Is This Engineering? Exploring Context Injection in a Probabilistic System

I wanted to design a simple experiment.

Pick a concrete, non-toy problem. Draft a short set of rules. Apply them to different AI agents. See whether the behavior shifted in reproducible ways.

For example:

If I tell one agent, “Ask clarifying questions before proposing changes,” and I tell another agent the same thing — do they both reliably adopt that posture? Or does the behavior drift? If I reset the session, does it hold?

The goal wasn’t to prove anything. It was to see whether I could reduce variance.

The structure felt familiar. It followed an engineering pattern:

- Define a behavioral goal.
- Draft constraints.
- Control inputs.
- Reset the environment.
- Observe output.
- Look for repeatability.

That’s how we stabilize unreliable systems. That’s how we probe unknown substrates. That’s how we move from intuition to something operational.

And yet, as soon as I framed it that way, something felt off.

The system I’m interacting with is probabilistic.

Identical prompts don’t guarantee identical outputs.

Context alters the conditional distribution of responses.

Reproducibility isn’t binary — it’s statistical.

If I add a short contract to the top of a ticket, the model shifts.

If I remove it, it shifts again.

If I move one sentence, sometimes the posture changes.

Nothing is broken.

But nothing is fixed, either.

So what does repeatability mean here?

What does control mean?

What does “engineering” mean?

The friction wasn’t about AI hype. It wasn’t about whether the tool is good or bad. It was about classification.

If I define rules, inject them, reset the session, and test transferability across agents — am I modeling a system? Or am I just steering text?

In deterministic systems, exploration and engineering feel distinct. You explore first, then formalize, then implement. The boundary is clearer.

In probabilistic systems, modeling requires interaction. You explore to understand variance. You add structure to shape distributions. You observe drift and adjust constraints. Exploration and control collapse into one motion.

That blur is the interesting part.

Context injection, stripped of drama, is simple: adding structured text alters the conditional probability of output. Tickets are context injection. Contracts are context injection. Process rules are context injection.

They don’t create belief. They shape distributions.

So if I’m trying to improve outputs by shaping those distributions — narrowing the behavioral range under reset conditions — what exactly am I doing?

It still looks like engineering. It still follows an engineering pattern. But it doesn’t look like the engineering I’m used to. The substrate behaves differently.

Maybe the discomfort isn’t about legitimacy. Maybe it’s about adaptation.

When the system produces distributions instead of certainties, engineering doesn’t disappear. But it stops looking like tightening bolts. It starts looking like shaping probability mass.

I don’t have a clean answer yet.

I know I was trying to reduce variance.

I know I was trying to create transferable constraints.

I know the structure followed an engineering pattern.

Whether that's "engineering" in the traditional sense feels less important than the fact that the work itself feels structurally familiar — even if the substrate doesn't.